初赛模拟卷

一、单项选择（共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项）

1. 下列不是 linux 命令的是（　）

A. ps　　B. usd

C. cp　　D. cd

2. （　）是目前互联网上常用的 E-mail 服务协议

A. POP3　　B. FTP

C. HTTP　　D. Telnet

3. 在计算机中，防火墙的作用是（　）

A. 防止火灾蔓延　　B. 防止网络攻击

C. 防止计算机死机　　D. 防止使用者误删数据

4. （　）就是把一个复杂的问题分成两个或更多的相同类似的子问题，再把子问题分解成更小的子问题……直到最后的子问题可以简单地直接求解。而原问题的解就是子问题的并。

A. 贪心　　B. 动态规划

C. 分治　　D. 搜索

5. 如果平面上任取 n 个整点（横纵坐标均为整数），其中一定存在两个点，它们连线的中点是整点，则 n 至少是（　）

A. 3　　B. 4

C. 5　　D. 8

6. 阅读下面的函数，当传入的三个参数为下列哪个时，函数能正常地获得返回值。（　）

```
int dfs(int a, int b, int c) {
    if(a == b) return 1;
    int newa = b * b / a, newb = c * c / b, newc = a * a / c;
    return dfs(newa, newb, newc);
}
```

A. a = 0, b = 1, c = 2　　B. a = 1, b = 2, c = 3

C. a = -1, b = 0, c = 1　　D. a = 3, b = 4, c = 5

7. 由 3 个 a,5 个 b 和 2 个 c 构成的所有字符串中，包含子串为 abc 的共有（　）个

A. 40320　　B. 39600

C. 840　　D. 780

8. 由 4 个不同的点构成的简单无向连通图的个数是（　）

A. 32　　B. 41

C. 35　　D. 38

9. 一棵二叉树前序遍历为 ABDECFGH，后序遍历为 EDBGFHCA，以下不可能是其中序遍历的是（　）

A. DEBAFGCH　　B. BEDAFGCH

C. EDBAGFCH　　D. DBEAFGCH

10. 数字 319 和数字 377 的最大公约数是（　）

A. 33　　B. 27

C. 29　　D. 21

11. 前缀表达式- + * 4 + 2 3 1 5 的值为（　）

A．16　　B．19

C．17　　D．15

12.以下关于二叉树性质中，正确的描述的个数有（　）

① 包含 n 个结点的二叉树的高度至少为 $ceil(log_2 n)$

② 在任意一棵非空二叉树中，若叶子结点的个数为 n_0，出度为 2 的结点数为 n_2，则 $n_0=n_2+1$

③ 深度为 k 的二叉树至多有 2^k 个结点

④ 没有一棵二叉树的前序遍历序列与后序遍历序列相同

A．0　　B．1

C．2　　D．3

13.将 2 个相同的红球，1 个蓝球，1 个白球放到 10 个编号不同的盒子中去，每个盒子最多放一个球，有多少种放法（　）

A．5040　　B．420

C．2520　　D．1260

14．设 A 和 B 是两个长度为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组，请问任何以元素比较作为基本运算的归并算法，在最坏情况下至少要做多少次比较？（　）

A．n^2　　B．nlogn

C．2n　　D．2n - 1

15．G 是一个非连通无向图（没有重边和自环），共有 45 条边，则该图至少有（　）个顶点。

A．8　　B．9

C．10　　D．11

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题选正误；除特殊说明外，判断题每题 2 分，选择题每题 3 分，共 40 分）

（一） 阅读下列程序，回答问题。

```
01  #include <bits/stdc++.h>
02  using namespace std;
03  const int maxn = 110;
04  int n, k, w[maxn]; double v[maxn];
05  struct node {
06      double pt; int total;
07      bool operator < (const node &temp) const {
08          return pt < temp.pt;
09      }
10  };
11  priority_queue<node> pq;
12  int main() {
13      cin >> n >> k; double ans = 0;
14      for(int i = 1;i <= n; ++i) cin >> v[i];
15      for(int i = 1;i <= n; ++i) cin >> w[i];
16      for(int i = 1;i <= n; ++i)
17          pq.push({v[i] / w[i], w[i]});
18      while(k > 0 && pq.size()) {
19          node u = pq.top(); pq.pop();
20          ans += min(k, u.total) * u.pt;
21          k -= min(k, u.total);
22      }
23      cout << ceil(ans) << '\n';
24      return 0;
25  }
```

假设所有输入的数据均为[0, 100]之间的自然数，回答下列问题。

- 判断题

16. 存在可以被构造出的输入数据，使得程序无法正确运行。（ ）

17. 对于优先队列 pq，当其非空时，若取出 pq 中的 top 处的元素，该元素必然是优先队列 pq 中 pt 属性较小的元素。（ ）

18. 当程序正常运行结束时，ans 一定小于等于$\sum v[i](1 \le i \le n)$。（ ）

- 选择题

19. 当 n = 5, k = 7, 数组 v[i]为[1, 2, 3, 4, 5], w[i]为[5, 4, 3, 2, 1]时，程序的输出结果为（ ）

A. 13　　B. 2

C. 12　　D. 1

（二） 阅读下列程序，回答问题。

```
01  #include <bits/stdc++.h>
02  using namespace std;
03  int main() {
04      int cnt[50]; memset(cnt, 0, sizeof cnt);
```

```
05      string str; cin >> str;
06      for(int i = 0;i < str.size(); ++i)
07          cnt[str[i] - 'a']++;
08      int ans = 0; bool judge = false;
09      for(int i = 0;i <= 25; ++i) {
10          if(cnt[i] % 2 == 0) ans += cnt[i];
11          else ans += cnt[i] - 1, judge = true;
12      }
13      if(judge) ++ans;
14      cout << ans << endl;
15      return 0;
16  }
```

假设输入的字符串仅包含小写字母，且长度不超过 10^5，完成下列问题。

- 判断题

20. 将第 09 行的 `i <= 25` 改为 `i <= 30`，不影响程序的结果。（　）

21. 当第 13 行中的 judge 为 true 时，求出的 ans 为偶数。（　）

- 选择题

22. 当输入的字符串为 helloworld 时，输出的答案为（　）

A. 7　　B. 5

C. 8　　D. 6

（三） 阅读下列程序，回答问题。

下面的代码的输入以下列方式给出：

第一行两个正整数 n，m,表示将给出一个长度为 n 的序列，并对其进行 m 次操作。其中，n 和 m 的范围在[1，1e6]之间。

第二行 n 个正整数，表示一个序列。

接下来 m 行，每行表示一种操作。

操作 1 给出的方式是 1 l r v，其中保证 1 <= l <= r <= n，1 <= v <= 1e9，表示对[l，r]这个区间做某种和 v 有关的操作。

操作 2 给出的方式是 2 l r，其中保证 1 <= l <= r <= n，表示对[l，r]这个区间做某种询问操作。

```
01  #include <cstdio>
02  #include <iostream>
03  #include <algorithm>
04  #include <cmath>
05  using namespace std;
06  const int maxn = 1e6 + 10;
07  long long tree[maxn * 4], num[maxn];
08  void build(int u, int l, int r) {
09      if(l == r) {
10          tree[u] = num[l] - num[l - 1];
11          return;
12      }
13      int mid = (l + r) / 2;
14      build(u * 2, l, mid);
```

```
    build(u * 2 + 1, mid + 1, r);
    tree[u] = min(tree[u * 2], tree[u * 2 + 1]);
    return;
}
void update(int u, int l, int r, int pos, long long v) {
    if(l == r) {
        tree[u] -= v;
        return;
    }
    int mid = (l + r) / 2;
    if(pos <= mid) update(u * 2, l, mid, pos, v);
    else update(u * 2 + 1, mid + 1, r, pos, v);
    tree[u] = min(tree[u * 2], tree[u * 2 + 1]);
    return;
}
long long query(int u, int l, int r, int L, int R) {
    if(L <= l && r <= R) return tree[u];
    int mid = (l + r) / 2; long long t = 1e9;
    if(L <= mid) t = min(t, query(u * 2, l, mid, L, R));
    if(R > mid) t = min(t, query(u * 2 + 1, mid + 1, r, L, R));
    return t;
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n, m; cin >> n >> m;
    for(int i = 1;i <= n; ++i)
        cin >> num[i];
    build(1, 1, n);
    for(int i = 1;i <= m; ++i) {
        int opt; cin >> opt;
        if(opt == 1) {
            int l, r; long long v;
            cin >> l >> r >> v;
            update(1, 1, n, l, -v);
            if(r + 1 <= n) update(1, 1, n, r + 1, v);
        } else {
            int l, r;
            cin >> l >> r;
            if(l == r) {
                cout << "Yes\n";
            } else {
                long long ans = query(1, 1, n, l + 1, r);
                cout << (ans >= 0 ? "Yes\n" : "No\n");
```

```
            }
        }
    }
    return 0;
}
```

- 判断题

23. 操作 1 是对区间进行减法操作，表示将区间[l, r]统一减去 v。（ ）

24. 当进行操作 2 时，若区间长度为 1，无论如何都会输出一行 Yes。（ ）

25. 程序中 32 行的 long long t = 1e9,改为 long long t = 0 仍然能正常求解。（ ）

- 选择题

26. (4 分) 操作 2 判断的是对于区间[l, r]，当区间满足（ ）时，程序会输出 Yes。

A. 单调增加　　B. 单调减小

C. 单调不增　　D. 单调不减

（四） 阅读下列程序，回答问题。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll l, r, f[25][15][15], a[25];
ll dfs(ll pos, bool limit, bool lead, ll pre1, ll pre2) {
    if (!pos) {
        return 1;
    }
    ll ans = 0;
    if (!limit && !lead && f[pos][pre1 + 1][pre2 + 1] != -1) {
        return f[pos][pre1 + 1][pre2 + 1];
    }
    ll up = limit ? a[pos] : 9;
    for (ll i = 0; i <= up; ++i) {
        if (i != pre1 && i != pre2) {
            ans += dfs(pos - 1, limit && i == up, lead && !i, (!lead
|| i) ? i : -1, pre1);
        }
    }
    if (!limit && !lead) {
        f[pos][pre1 + 1][pre2 + 1] = ans;
    }
    return ans;
}
ll solve(ll x) {
    ll cnt = 0;
    while (x) {
        a[++cnt] = x % 10;
        x /= 10;
    }
```

```
30      memset(f, -1, sizeof(f));
31      return dfs(cnt, 1, 1, -1, -1);
32  }
33  int main() {
34      cin >> l >> r;
35      cout << solve(r) - solve(l - 1);
36      return 0;
37  }
```

假设输入的 l 和 r 都是不超过 10^{18} 的自然数，本程序目标是统计位于区间[l, r]中具有某种性质的正整数的数量。据此完成下列问题。

- 判断题

27. 将第 07 行改为 `return 0;`程序统计出的数量不会改变。（　）

28. (3 分) 第 15 行的作用是保证对所有被统计到答案的数字中，不会出现连续的三个相等的正整数。（　）

29. 当输入的 l 和 r 相等时，输出答案必定为 1。（　）

- 单选题

30. 若输入数据为 1 100，则输出的答案为（　）

A. 100　　B. 90

C. 70　　D. 80

31. (4 分) 若输入数据为 1 1000，则输出的答案为（　）

A. 991　　B. 688

C. 738　　D. 832

三、完善程序（单选题，每小题 3 分，共计 30 分）

（一） 现在要求你求出满足下列要求的序列的数量，答案对 10^9+7 取模。

① $a_0 = 0$

② $\forall i \in [1,n]$，满足 $a_i = a_{i-1} + x$ 或 $a_i = a_{i-1} + y$

③ $\forall i \in [1,n]$，满足 $a_i \bmod p \neq 0$

输入数据格式为第一行一个正整数 t($1 \le t \le 1000$)，表示有 t 组询问。

对每组询问给出一行四个正整数 n，p，x，y，以空格分开。

保证$1 \le \sum n \le 10^4, 1 \le p,x,y \le 10^9$。对每组询问你都需要回答一行正整数表示答案。

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cmath>
#include <cstring>
using namespace std;
const int maxn = 1e4 + 10;
const int mod = 1e9 + 7;
int n, dp[2 * maxn];
long long p, x, y;
int main( ) {
    int t; cin >> t;
    while(t--) {
        int ans = 0;
        cin >> n >> p >> x >> y;
        if(x == y) {
            bool judge = true;
            for(int i = 1;i <= n; ++i) if(_____①_____) judge = false;
            cout << _____②_____ << '\n';
            continue;
        }
        dp[maxn] = 1;
        for(int i = 1;i <= n; ++i) {
            for(int j = 0;j <= i; ++j) {
                int t = _____③_____; dp[maxn + t] = 0;
                if(_____④_____)
                    dp[maxn + t] = (_____⑤_____) % mod;
                if(i == n) ans = (ans + dp[maxn + t]) % mod;
            }
        }
        cout << ans << '\n';
        for(int i = -n; i <= n; ++i) dp[maxn + i] = 0;
    }
    return 0;
}
```

32. ①处应填（　）

A. x % p == 0　　B. x * i % p == 0
C. i % p == 0　　D. judge && (x % p == 0)

33. ②处应填（　）
A. judge ? 1 : 0　　B. judge ? n : 0
C. judge ? 0 : 1　　D. judge ? 0 : n

34. ③处应填（　）
A. j　　B. j + (i – j)
C. i - j　　D. j – (i – j)

35. ④处应填（　）
A. (x * i + y * j) % p == 0
B. (x * i + y * j) % p != 0
C. (x * j + y * (i – j)) % p == 0
D. (x * j + y * (i – j)) % p != 0

36. ⑤处应填（　）
A. dp[maxn + t + 1] + dp[maxn + t - 1]
B. dp[maxn + t + 1] - dp[maxn + t - 1]
C. dp[t + 1] + dp[t - 1]
D. dp[t + 1] - dp[t - 1]

（二） 现在你有一个包含 n 个不同的字符串的串集 S。现在询问你有多少个不同的长度为 m 的字符串，使得字符串中至少有一个连续子串，位于串集 S 中。由于答案可能很大，你需要对 10007 取模。

所有出现的字符均只考虑大写字母。

数据范围有$1 \le n \le 60, 1 \le m \le 100$，每个字符串的串长$\in [1,100]$

输入格式为第一行两个正整数 n,m，表示你拥有的串集大小为 n，你可以构造的字符串的长度为 m。

接下来 n 行，每行一个字符串，表示串集中的一个字符串。

提示：考虑用所有字符串的数量减去非法字符串的数量。

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 6000 + 10;
const int mod = 10007;
struct Aho_corasick{
    int ch[maxn][30], f[maxn], tot;
    bool endp[maxn];
    Aho_corasick() {
        memset(ch, 0, sizeof ch);
        memset(endp, 0, sizeof endp);
        memset(f, 0, sizeof f);
        tot = 0;
    }
    void insert(string &s) {
        int n = s.size(); int u = 0;
        for(int i = 0;i < n; ++i) {
            if(!ch[u][s[i] - 'A'])
                ch[u][s[i] - 'A'] = ++tot;
            u = ch[u][s[i] - 'A'];
        }
        _____①_____;
    }
    void getfail() {
        queue<int> q;
        for(int i = 0;i <= 25; ++i)
            if(ch[0][i])
                q.push(ch[0][i]);
        while(!q.empty()) {
            int u = q.front(); q.pop();
            for(int i = 0;i <= 25; ++i) {
                int v = ch[u][i];
                if(v) {
                    f[v] = ch[f[u]][i];
                    _____②_____
                    q.push(v);
```

```
                } else {
                    ch[u][i] = ch[f[u]][i];
                }
            }
        }
    }
} AC;
int quickpow(int x, int y) {
    int base = x, ans = 1;
    while(y) {
        if(y & 1) _____③_____
        base *= base;
        ans %= mod; base %= mod;
        y /= 2;
    }
    return ans;
}
int ans, dp[105][maxn];
int main() {
    int n, m; cin >> n >> m;
    ans = quickpow(26, m);
    for(int i = 1;i <= n; ++i) {
        string str; cin >> str;
        AC.insert(str);
    }
    AC.getfail();
    int max_state = AC.tot;
    dp[0][0] = 1;
    for(int i = 0;i < m; ++i) {
        for(int state = 0;state <= max_state; ++state) {
            for(int nxt = 0;nxt <= 25; ++nxt) {
                if(_____④_____) continue;
                _____⑤_____ += dp[i][state];
                _____⑤（两处⑤所填内容相同）_____ %= mod;
            }
        }
    }
    for(int state = 0; state <= max_state; ++state)
        ans = (ans + mod - dp[m][state]) % mod;
    cout << ans << '\n';
    return 0;
}
```

37. ①处应填（　）

A. endp[u] = 0　　B. endp[u] = 1

C. f[u] = 0　　D. f[u] = 1

38. ②处应填（　）

A. if(endp[f[u]]) endp[u] = 1;

B. if(endp[f[u]]) endp[v] = 1;

C. if(endp[f[v]]) endp[u] = 1;

D. if(endp[f[v]]) endp[v] = 1;

39. ③处应填（　）

A. ans *= base;　　B. base *= ans

C. ans += base　　D. base += ans

40. ④处应填（　）

A. AC.f[AC.ch[state][nxt]]

B. AC.endp[AC.ch[state][nxt]]

C. AC.f[AC.ch[nxt][state]]

D. AC.endp[AC.ch[nxt][state]]

41. ⑤处应填（　）

A. dp[i][AC.f[AC.ch[state][nxt]]]

B. dp[i + 1][AC.f[AC.ch[state][nxt]]]

C. dp[i][AC.ch[state][nxt]]

D. dp[i + 1][AC.ch[state][nxt]]